04

Lanzando excepciones

Transcripción

[00:00] Hemos venido nuevamente a nuestro IDE, a Eclipse, y vemos que aquí tenemos abierta la clase flujo. Ahora ya hemos visto cómo es el comportamiento de este método y sabemos que si sucede alguna cosa inesperada, un error inesperado, con catch aquí él va a atrapar las excepciones.

[00:25] Ahora vamos a llevar este caso a uno un poco más práctico ya del mundo real, y para esto nuevamente yo voy a hacer uso del proyecto anterior de herencia y polimorfismo de nuestro Byte Bank. Nuevamente ustedes no necesitan descargar el proyecto aún. Es solamente para fines ilustrativos, por así decirlo. Vamos a nuestra claseCuenta y venimos aquí a nuestro método saca. ¿Por qué?

[00:50] En nuestro método saca, recordando un poco lo que ya hablamos en la primera clase y lo que hablamos en su momento, cuando implementamos el método, ¿cuál era la idea leyendo el código? Si es que yo tengo saldo suficiente para sacar dinero, entonces resta el saldo del valor de dinero que yo tengo y retorna true, si no, retorna false.

[01:13] Pero ahora yo aquí no tengo una explicación totalmente cierta de a qué hace referencia el hecho que yo no pueda retirar dinero. Por ejemplo, yo como cliente yo voy al cajero automático, y si yo quiero retirar dinero, él simplemente va a decir: "no puedes". Y yo: "¿Por qué no puedo?" Y el cajero va a decir solamente: "no, simplemente no puedes".

[01:37] Esas cosas no pueden ocurrir. ¿Por qué? Porque si ocurre una queja de cliente, si yo como cliente me quejo con el banco: "Byte Bank, ¿sabes qué? Yo no puedo sacar dinero de mi cuenta". Si en algún caso ellos necesitan ver los logs del sistema del aplicativo, ellos van a tener la misma respuesta que yo tuve: "Simplemente no puedes sacar tu dinero".

[02:00] Y ningún tipo de indicio o de información de por qué es que no puedo sacar el dinero de mi cuenta. Y ya hemos visto que motivos para no poder sacar el dinero de mi cuenta hay muchos, como por ejemplo es domingo, no puedo hacer operaciones día domingo, no tengo saldo, mi cuenta está bloqueada, mi tarjeta el cartón ya está bloqueado y una infinidad de razones.

[02:24] Y entonces quizás yo podría completar esto, pues con otro if, de repente, a ver, escribiendo un seudo código, escribiendo acá else if, por ejemplo, cuentaBloqueada, if cuentaBloqueada, entonces yo podría comenzar, digamos, a catalogar mis errores, digamos yo puedo ponerle a cuenta bloqueada, tipo de error menos 1. Es domingo menos 2, no tiene saldo menos 3 y así crear un montón de de estos ifs.

[03:03] Crear un montón de códigos que solamente quizás en el momento yo voy a conocer, que de aquí al siguiente año no van a tener el menor sentido y va a complicar mucho más el trabajo de mantener este sistema y recordemos que nuestro trabajo como programadores al mismo tiempo es hacer nuestro código lo más mantenible para otros programadores, para que ellos lleguen y encuentren el código limpio y consigan puede dar un mantenimiento adecuado a nuestro sistema.

[03:31] Ahora en este punto, ya no sé qué puedo hacer. Ya sé que sí yo tengo demasiados ifs, ya es un síntoma de que necesito refactorizar el código. ¿Para qué son las excepciones? Justo para atacar este caso. ¿Por qué? Porque por ejemplo sabemos que Java es un lenguaje orientado a objetos. Y las excepciones también son objetos. ¿Por qué?

[03:54] Volvemos aquí a nuestro flujo. Si nosotros vemos aquí lo que estamos haciendo con nuestra variable excepcion, nosotros tenemos métodos como printStackTrace y un método getMessage. Entonces, las excepciones también son objetos y nosotros podríamos crear nuestras propias excepciones también.

[04:15] Ahora, si las excepciones son objetos, ¿será que yo puedo ir creando excepciones así, en el transcurso del Código? Vamos a ver eso ahorita. Y vamos a ir aquí al código, a nuestro flujo. Sí, estamos en la clase correcta. Y aquí vamos a hacer algo interesante en Metodo 2. Este for que está aquí, yo lo voy a borrar por el momento. ¿Por qué?

[04:50] Vamos a crear nuestra propia excepción, nuestra propia bomba, por ejemplo. Vimos que la excepción era ArithmeticException, entonces voy a decirle ArithmeticException, y vemos que él aquí ya me sugiere ArithmeticException. Yo puedo crear un objeto ArithmeticException, él aquí ya me declaró un tipo de objeto ArithmeticException.

[05:17] Y si yo quiero, digamos, crear una referencia a ese objeto, entonces le puedo dar un new, new ArittmeticException, punto y coma al final. El ahí ya va a crear un objeto directamente en la memoria de Java, en la memoria head. Pero yo este objeto también lo puedo referenciar, porque en el catch yo podía referenciar mi objeto dentro de una variable.

[05:40] Entonces yo puedo poner aquí de la misma forma ArithmeticException, ae es igual a new ArithmeticException. ¿Conseguí crear mi propia excepción? Sí, aquí yo he creado mi propia excepción. Un ArithmeticException,creado por mí, mi propia bomba, yo mismo la he hecho.

[06:01] Y si yo ejecuto método 2 ahora, bueno, mejor dicho, si ejecuto main para que ejecute todo, vamos a ver qué es lo que sucede con mi excepción, que yo he creado, voy a guardar aquí y voy a ejecutar aquí y vemos aquí en el output de la consola que él hizo inicio main, inicio método 1, inicio método 2, fin método 2, fin método 1 y fin de main.

[06:25] Y ahora yo me pregunto: ¿dónde está mi excepción? Yo aquí he creado mi excepción explícitamente, es un ArithmeticException, yo lo he hecho. ¿Por qué es que esta bomba que yo he creado en el código no fue lanzada? Vamos a ver la respuesta a esa pregunta en el siguiente video.